Security System

EE 211 Final Project

27 April 2021

Jeremy Meade, Ali Jamal

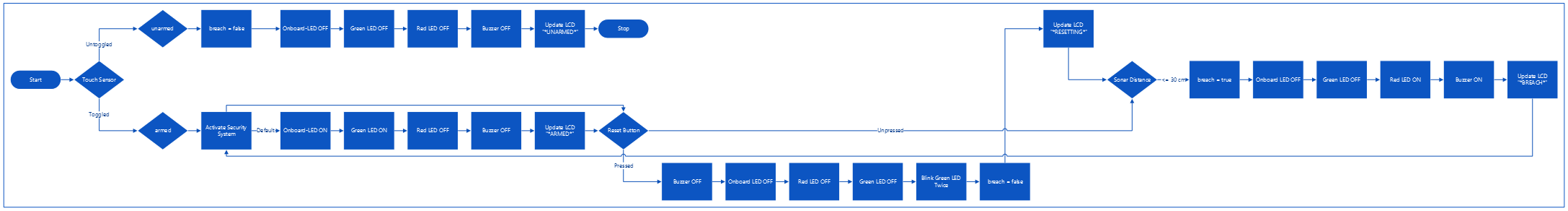
**Interface Design:**

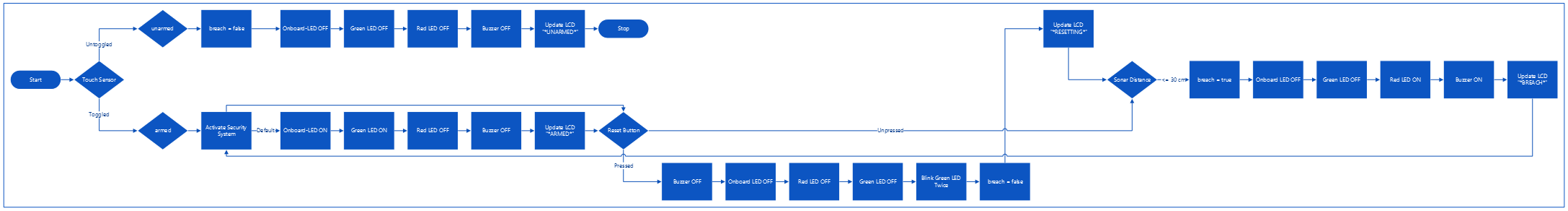
Devices/Components used:

Arduino Uno and a variety of Sunfounder components including: ultrasonic sonar sensor, liquid crystal display, dual-color led, push button, touch sensor, and active buzzer.

**Software:**

**Flowchart**

****

****

**Code**

#include <Wire.h> //include the header file for LCD

#include <LiquidCrystal\_I2C.h> //include the header file for the LCD

#include <NewPing.h> //include the header file for the ultrasonic sensor

#define buttonPin 7 //button attached to pin 7

#define ledPin 13 //built-in led to pin 13

#define redPin 11 //red pin for dual-led to pin 11

#define greenPin 10 //green pin for dual-led to pin 10

#define buzzerPin 4 //active buzzer to pin 4

#define touchPin 12 //touch-switch to pin 2

#define TRIGGER\_PIN 2 // Arduino pin tied to trigger pin on the ultrasonic sensor.

#define ECHO\_PIN 3 // Arduino pin tied to echo pin on the ultrasonic sensor.

#define MAX\_DISTANCE 400 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

bool armed = 1; //default active system

bool breach = 0; //default no breach system status

char displayTitle[] = "Security System"; //title for LCD

char armedStatus[] = " \*ARMED\* "; //armed status for LCD

char unarmedStatus[] = " UNARMED "; //unarmed status for LCD

char breachStatus[] = "\*BREACH\* "; //breach status for LCD

char resetStatus[] = "RESETTING"; //reset status for LCD

LiquidCrystal\_I2C lcd(0x27,16,2); //definition of "lcd" as the LCD display with 16 columns and 2 rows

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.

void setup() {

Serial.begin(9600); //initialize the serial monitor

pinMode(buttonPin,INPUT); //set buttonPin as INPUT

pinMode(ledPin, OUTPUT); //set ledPin as INPUT

pinMode(redPin, OUTPUT); //set redPin as INPUT

pinMode(greenPin, OUTPUT); //set greenPin as INPUT

pinMode(buzzerPin, OUTPUT); //set buzzerPin as INPUT

pinMode(touchPin, INPUT); //set touchPin as INPUT

lcd.init(); //initialize LCD display

lcd.backlight(); //turn on LCD backlight

}

void loop() {

bool buttonState = digitalRead(buttonPin); //read the value of pin 7

bool touchState = digitalRead(touchPin); //read the value of pin 2

int distance = soundDistance();

if (touchState==0){ //sets system to be armed or unarmed depending on value of touch-switch

armed = 1;

}

else{

armed = 0;

}

if (armed==1){ //activates security system

securitySystem(distance, buttonState);

}

else{ //deactivates security system

breach = 0;

digitalWrite(ledPin, LOW);

analogWrite(greenPin, 0);

analogWrite(redPin, 0);

digitalWrite(buzzerPin, HIGH);

unarmedSecurityDisplay();

}

}

void securitySystem(int distance, bool button){

if (distance <= 30 and distance > 0){ //tests for door open and at odd angles, sensor would report the ostensible distance of 0cm

breach = 1;

digitalWrite(ledPin, LOW);

analogWrite(greenPin, 0);

analogWrite(redPin, 255);

digitalWrite(buzzerPin, LOW);

breachSecurityDisplay();

}

else if (button==LOW){ //resets system

digitalWrite(buzzerPin, HIGH);

digitalWrite(ledPin, LOW);

analogWrite(redPin, 0);

analogWrite(greenPin, 0);

resetSecurityDisplay();

delay(250);

analogWrite(greenPin, 255);

delay(250);

analogWrite(greenPin, 0);

delay(250);

analogWrite(greenPin, 255);

delay(250);

lcd.clear();

breach = 0;

}

else if (breach==0){ //default system status

digitalWrite(ledPin, HIGH);

analogWrite(greenPin, 255);

analogWrite(redPin, 0);

digitalWrite(buzzerPin, HIGH);

armedSecurityDisplay();

}

}

void armedSecurityDisplay(){ //display when system is armed

lcd.setCursor(0,0);

lcd.print(displayTitle);

lcd.setCursor(4,1);

lcd.print(armedStatus);

}

void unarmedSecurityDisplay(){ //display when system is unarmed

lcd.setCursor(0,0);

lcd.print(displayTitle);

lcd.setCursor(4,1);

lcd.print(unarmedStatus);

}

void breachSecurityDisplay(){ //display when system detects a breach

lcd.setCursor(0,0);

lcd.print(displayTitle);

lcd.setCursor(4,1);

lcd.print(breachStatus);

}

void resetSecurityDisplay(){ //display when system is resetting

lcd.setCursor(0,0);

lcd.print(displayTitle);

lcd.setCursor(4,1);

lcd.print(resetStatus);

}

int soundDistance(){

delay(100); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

unsigned int ultraSonic = sonar.ping(); // Send ping, get ping time in microseconds (uS).

unsigned int distance = ultraSonic / US\_ROUNDTRIP\_CM;

Serial.print("Ping: ");

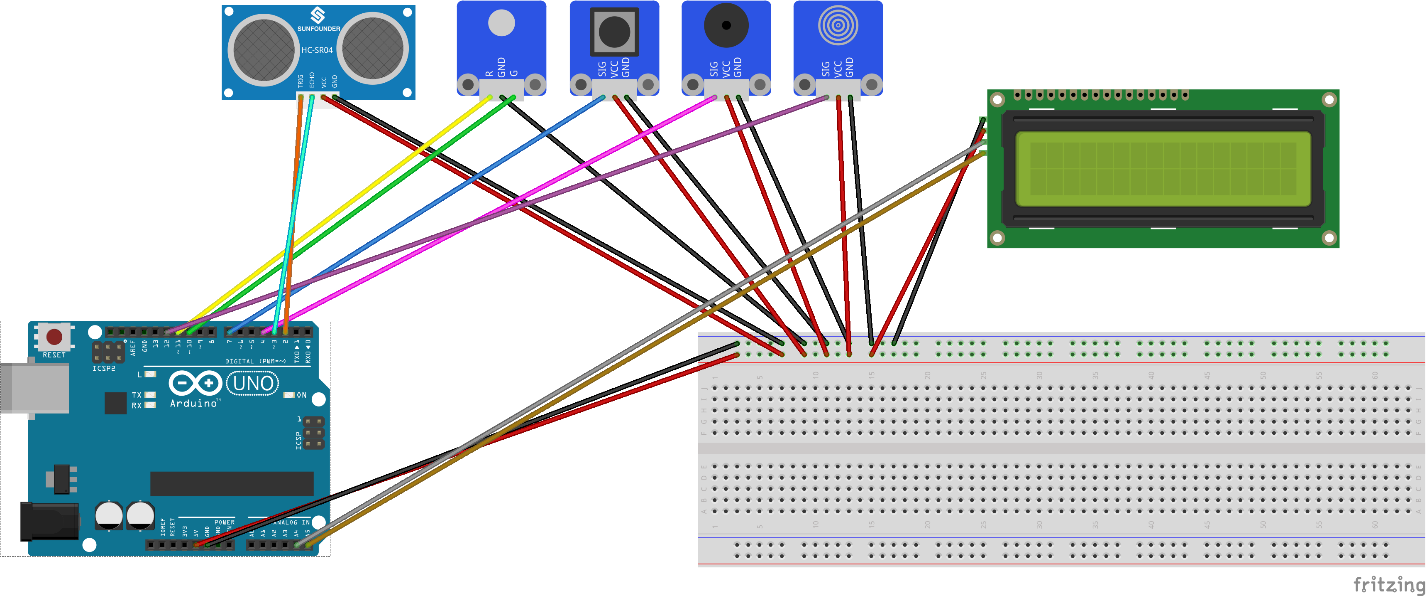
Serial.print(distance); // Convert ping time to distance in cm and print result (0 = outside set distance range)

Serial.println("cm");

return distance;

}

**Circuit Diagram:**

****

**Analysis:**

Operation of the system was tested by positioning the ultrasonic sensor facing a doorway. When the door was opened, the ultrasonic sensor would register the distance between itself and an object to be smaller; therefore, a door being open. When the door is opened, and the system is armed, the system will detect the breach by way of the ultrasonic sensor, change the LCD to say \*BREACH\*, the dual-color LED will turn red, the active buzzer will turn on, and the on-board LED will turn off. The system can be reset by either pressing the button - displaying \*RESETTING\* on the LCD, turning the dual-color LED green, turning the buzzer off, and returning to the state of listening for the ultrasonic sensor to detect a door being opened; or by pressing the touch switch, thus unarming the system.

In conclusion, this project was a good test of our abilities, coding or otherwise. The security system could be improved markedly by using a hall sensor attached to the door frame, and a magnet attached to the door. This would have vastly improved the issue of false positives that the ultrasonic sensor produces. Of course, more hardware could be added such as cameras, motion sensors, or sound sensors. Also, facial recognition could be implemented in the software. Additionally, uploading to the cloud, creating a mobile app for notifying of detected break-ins or faces. In the future, we would like experience more real-world implications of software development, A.K.A, using software to interact with the physical world.